(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) **EP 2 360 918 A1**

(12) **EUROPEAN PATENT APPLICATION**

(43) Date of publication:
**24.08.2011 Bulletin 2011/34**

(51) Int Cl.:
***H04N 7/16*** *(2011.01)* ***H04L 9/08*** *(2006.01)*

(21) Application number: **10196221.5**

(22) Date of filing: **21.12.2010**

(84) Designated Contracting States:
**AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR**
Designated Extension States:
**BA ME**

(30) Priority: **26.01.2010 EP 10151677**

(71) Applicant: **IRDETO B.V.**
**2132 LS Hoofddorp (NL)**

(72) Inventors:
- **Benedetti, Ettore**
  **2132 LS, Hoofddorp (NL)**
- **Van Foreest, Arnoud Evert**
  **2132 LS, Hoofddorp (NL)**

(74) Representative: **van Looijengoed, Ferry Antoin Theodorus et al**
**De Vries & Metman**
**Overschiestraat 180**
**1062 XK Amsterdam (NL)**

# (54) Computational efficiently obtaining a control word in a receiver using transformations

(57) The invention provides a receiver, a smartcard and a conditional access system for securely obtaining a control word using an entitlement transform tree, wherein intermediate results are cached to improve computational efficiency.

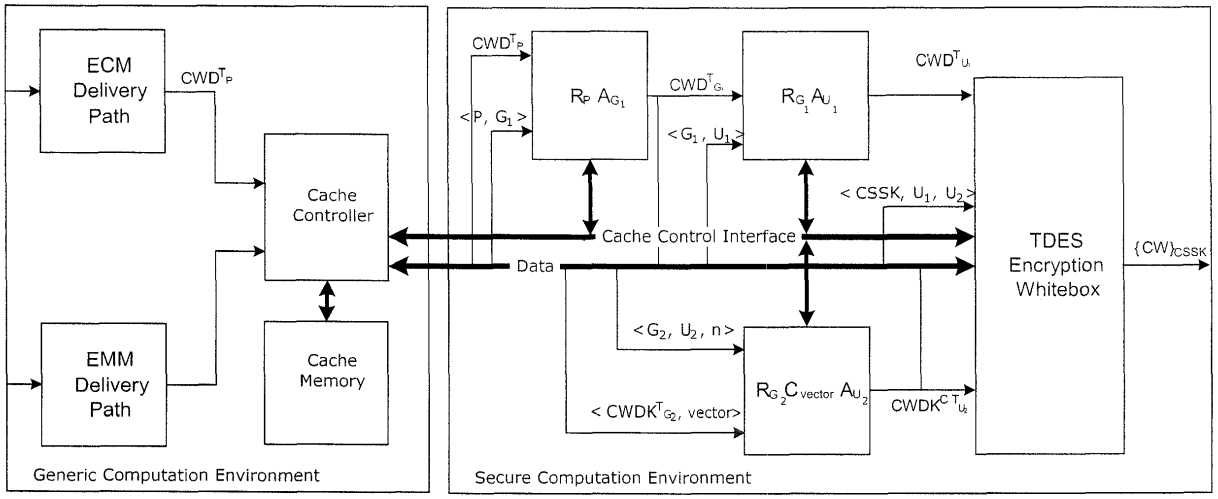


Fig.9

EP 2 360 918 A1

## Description

### FIELD OF THE INVENTION

**[0001]** The present invention relates to a receiver, a smartcard, a conditional access system and a method for computational efficiently obtaining a control word using transformation functions.

### BACKGROUND

**[0002]** Conditional Access systems, such as Pay-TV systems, are known that use software tamper resistance to protect key storage and entitlement processing steps in a digital TV receiver.

**[0003]** Software tamper resistance technology uses basic primitives to obscure software code transformations. Examples of basic primitives are "Apply", "Remove" and "Condition". Fig. 1A, Fig. 1B and Fig. 1C show block diagrams of an apply primitive A, a remove primitive R and a condition primitive C, respectively. The apply primitive typically uses a function $A(D, S) = A_s(D) = D^{TS}$ to transforms a data element D according to a parameter seed S. The remove primitive typically uses a function $R(D^{TS}, S) = R_s(D^{TS}) = D$ to reverse the transformation of a data element D based on a seed S. The conditional primitive typically uses a function $C(D_1,D_2)=CD_1(D_2)=D^{CS}$, wherein the output is a correlation of the two inputs.

**[0004]** The seed S can be constructed from a mixture of multiple data elements. This makes it difficult to extract the individual data elements from the seed. The parameter mixing functions are typically denoted as f(A,B)=<A,B>. The function result <A,B> is called the compound of A and B. Hereinafter, seeds and compounds are both referred to as "seeds".

**[0005]** The primitives are typically combined when implementing key management functions in a Conditional Access system. The combination of primitives results in a new function wherein the individual primitives are no longer identifiable. Known examples of combinations of primitives are a combination of remove and apply primitives and a secure correlation of compounds.

**[0006]** Fig. 1D shows an instance of a combination of remove and apply primitives. The transformation uses a compound <P,S> in a combined remove and apply operation. The function $R_pA_s$ modifies the input data by replacing a transformation using the seed P with a transformation using the seed S, i.e. $Data^{TP}$ is transformed into $Data^{TS}$.

**[0007]** Fig. 1E shows an instance of a secure correlation of compounds. It is typically used for conditional entitlement processing and comprises a combination of the basic primitives apply, remove and condition. The conditional function can be combined with remove and apply blocks $R_pA_s$ of Fig. 1D to perform a secure correlation of compounds.

**[0008]** A method and a receiver for conditional entitlement processing and obtaining a control word CW is described in related European patent application no. 09155007.9, which is hereby incorporated by reference into this application.

**[0009]** Fig.2 shows an example of a split key delivery as described in EP09155007.9. In Fig.2 a CW is generated from three subkeys $CW_1$, $CW_2$ and $CW_3$. The subkeys $CW_1$, $CW_2$ and $CW_3$ are distributed under protection of seeds P, G and U, respectively. Hereto $CW_1$ is distributed in a mathematically transformed form in transformation space P, $CW_2$ is distributed in a mathematically transformed form in transformation space G and $CW_3$ is distributed in a mathematically transformed form in transformation space U.

**[0010]** Fig.3 shows an example of CW processing in a receiver as described in EP09155007.9. In Fig.3 the CW is generated from subkeys and a membership check is performed. The processing is divided in two basic parts: a secure computation environment and a generic processing environment. Functional modules in the generic processing environment and the secure computation environment form an entitlement transform tree for transforming an input transformed CW, e.g. $CWD^{TP}$, into a CW encrypted using a receiver specific key, e.g. $\{CW\}_{CSSK}$. The generic processing environment deals with the external interfaces such as storage, data communication and user interaction. The secured computation environment deals with the processing of keys and/or seeds.

**[0011]** An ECM Delivery Path is used for the reception of entitlement control messages (ECM) from a head-end system. The ECM comprises an encrypted or transformed CW. An EMM Delivery Path is used for the reception of entitlement management messages (EMM) from the head-end system. The EMM comprises keys or seeds for decrypting or transforming the encrypted or transformed CW.

**[0012]** The software tamper resistance primitives in the secure computation environment have inputs and outputs that are not useful to an attacker if intercepted. The remove operation on the transformed control word $CWD^{TP}$ requires value P, which is received in a compound <P, $G_1$>, thus tied with $G_1$. $G_1$ is distributed in a compound <$G_1$, $U_1$>, thus tied with $U_1$. After the two Remove/Apply operations $R_PA_{G1}$ and $R_{G1}A_{U1}$, the obtained transformed control word $CWD^{TU1}$, is input to a TDES Encryption Whitebox module for the generation of an encrypted CW that can be processed by the receiver. The resulting CW is encrypted using a receiver specific key such as a chip set session key CSSK. The CSSK is typically provided in one of the entitlement messages. The CSSK, U1 and U2 values are typically provided to the TDES Encryption Whitebox as a compound <CSSK, $U_1$, $U_2$>.

**[0013]** The conditional entitlement processing of Fig.3 uses a secure correlation function $R_{G2}C_{vector}A_{U2}$ to implement a group membership check. A result of the correlation computation is a Control Word Difference Key CWDK in transformation space $U_2$, i.e. $CWDK^{CTU2}$. $CWDK^{CTU2}$ and $CWD^{TU1}$ are subkeys used in the calculation of the CW in the TDES Encryption Whitebox.

**[0014]** Subkeys, such as $CW_1$, $CW_2$ and $CW_3$ of Fig.2 and $CWDK^{CTU2}$ and $CWD^{TU1}$ of Fig. 3, may have different life spans. As an example $CW_1$ may change on a regular basis such as every 10 seconds, $CW_2$ may change on a sporadic basis measured in days and $CW_3$ may change very seldom measured in months.

**[0015]** Known software tamper resistant conditional entitlement processing technologies for the obtainment of CWs from transformed subkeys do not take into account the different life spans of subkeys. As a consequence all intermediate operations in the conditional entitlement processing are always performed in order to obtain the CW. The execution of each intermediate operation is expensive in terms of processor cycles.

**[0016]** There is a need to reduce the number of computations in software tamper resistant conditional entitlement processing technologies, especially in devices wherein processing capabilities are limited, while not adversely affecting the tamper resistance of the implementation.

## SUMMARY OF THE INVENTION

**[0017]** It is an object of the invention to provide an improved software tamper resistant conditional entitlement processing technology for the obtainment of CWs, wherein computational efficiency is increased.

**[0018]** According to an aspect of the invention a receiver is proposed for securely obtaining a control word. The receiver comprises a first memory configured for storing a transform function. The transform function is configured to receive a transformed control word and a seed and to migrate the transformed control word from an input transform space to an output transform space. Hereby the transform function obtains the control word using a mathematical transformation under control of the seed. The receiver further comprises a cache memory and a cache control module. The cache control module is configured to intercept the transformed control word and the seed. The cache control module is further configured to search in the cache memory for the control word matching the transform function, the transformed control word (x) and the seed (y). The cache control module is further configured to, if the control word is found in the cache memory, provide the control word to an output of the transform function thereby bypassing the transform function. The cache control module is further configured to, if the control word is not found in the cache memory, provide the control word and the seed to the transform function, obtain the control word from the transform function and store the control word associatively with the transform function, the transformed control word (x) and the seed (y) in the cache memory.

**[0019]** According to an aspect of the invention a method is proposed for securely obtaining a control word in a receiver. The receiver comprises a first memory configured for storing a transform function. The method comprises the step of receiving a transformed control word and a seed. The method further comprises the step of intercepting in a cache control module the transformed control word and the seed. The method further comprises the step of searching in a cache memory for the control word matching the transform function, the transformed control word and the seed. The method further comprises the step of, if the control word is found in the cache memory, providing the control word to an output of the transform function thereby bypassing the transform function. The method further comprises the steps of, if the control word is not found in the cache memory, providing the control word and the seed to the transform function, migrating in the transform function the transformed control word from an input transform space to an output transform space to obtain the control word using a mathematical transformation under control of the seed, and storing the control word associatively with the transform function, the transformed control word and the seed in the cache memory.

**[0020]** Thus, the control word in the output transform space is not computed by the transform function if the expected result is, based on the input to the transform function, available in the cache memory. Hereby the computational efficiency in obtaining the control word is increased.

**[0021]** The output transform space may be a cleartext transform space, resulting in the control word being in cleartext. The resulting cleartext control word may be encrypted after being obtained. The output transform space may be any other transform space, requiring a further transformation of the control word to obtain the control word in the cleartext transform space.

**[0022]** The embodiments of claims 2 and 10 advantageously enable subsequent transformations in a sequence of transform functions and/or combining of transformed subkeys in a tree of transform functions, wherein intermediate results are cached for computational efficiency.

**[0023]** The embodiment of claim 3 advantageously enables the end result of the computations, i.e. the clear text control word or the encrypted control word, to be used in the receiver for descrambling content.

**[0024]** The embodiment of claim 4 advantageously enables obfuscation of computer code and functional behaviour of the transform functions, making it more difficult to obtain information about the control word during the mathematical transformations. Advantageously, the cached intermediate results can be stored in conventional non-obfuscated memory, making the cache memory easier and cheaper to implement.

**[0025]** The embodiment of claim 5 advantageously enables caching functionality with only a single cache control module in the generic computation environment.

**[0026]** The embodiment of claim 6 advantageously enables the cache control functionality to be implemented in the secure computation environment, leaving only the cache memory part of the caching in the generic computation environment. This results in less modifications of the generic computation environment for implementing the caching functionality.

**[0027]** The embodiment of claim 7 advantageously enables the caching functionality in conditional access systems using smartcards for the obtainment of control words.

**[0028]** The embodiment of claim 8 advantageously enables sharing of the smartcard in a network, wherein intermediate results can be cached in each receiver.

**[0029]** According to an aspect of the invention a smartcard is proposed for use in a receiver having one or more of the above described features. The smartcard comprises a first memory in a secure computation environment. The first memory is configured for storing a transform function. The transform function is configured to receive a transformed control word and a seed and to migrate the transformed control word from an input transform space to an output transform space. Hereby the transform function obtains a control word using a mathematical transformation under control of the seed. The transform function comprises a cache control module. The cache control module is configured to intercept the transformed control word and the seed. The cache control module is further configured to search in a cache memory of the receiver for the control word matching the transform function, the transformed control word (x) and the seed. The cache control module is further configured to, if the control word is found in the cache memory, provide the control word to an output of the transform function thereby bypassing the transform function. The cache control module is further configured to, if the control word is not found in the cache memory, provide the control word and the seed to the transform function, obtain the control word from the transform function and store the control word associatively with the transform function, the transformed control word and the seed in the cache memory.

**[0030]** Thus, caching functionality is enabled in conditional access systems using smartcards for the obtainment of control words. The control word in the output transform space is not computed by the transform function if the expected result is, based on the input to the transform function, available in the cache memory. Hereby the computational efficiency in obtaining the control word in the smartcard is increased.

**[0031]** The smartcard is typically implemented having a traditional form factor. Any other computing device implementing smartcard technology may be used as a smartcard instead, such as e.g. a PC running smartcard emulation software.

**[0032]** According to an aspect of the invention a conditional access system is proposed. The conditional access system comprises a head-end system and one or more receivers having one or more of the above described features. The head-end system is configured to transmit an entitlement control message and an entitlement management message to the receiver. The entitlement control message comprises the transformed control word. The entitlement management message comprises one or more seeds.

**[0033]** Thus, caching functionality is enabled in conditional access systems, wherein a transformed control word is provided by a head-end system to a receiver for transforming the transformed control word from an input transform space to an output transform space. The control word in the output transform space is not computed by the transform function if the expected result is, based on the input to the transform function, available in the cache memory. Hereby the computational efficiency in obtaining the control word is increased.

**[0034]** Hereinafter, embodiments of the invention will be described in further detail. It should be appreciated, however, that these embodiments may not be construed as limiting the scope of protection for the present invention.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0035]** Aspects of the invention will be explained in greater detail by reference to exemplary embodiments shown in the drawings, in which:

Fig. 1A shows a prior art block diagram of an apply primitive as used in software tamper resistance technology;
Fig. 1B shows a prior art block diagram of a remove primitive as used in software tamper resistance technology;
Fig. 1C shows a prior art block diagram of a condition primitive as used in software tamper resistance technology;
Fig. 1D shows a prior art block diagram of a combination of remove and apply primitives as used in software tamper resistance technology;
Fig. 1E shows a prior art block diagram of a secure correlation of compounds as used in software tamper resistance technology;
Fig.2 shows a prior art simplified split key delivery process;
Fig.3 shows a prior art split key delivery process in a receiver;
Fig.4 shows a transform function of an exemplary embodiment of the invention;

Fig.5 shows a transform function with caching of an exemplary embodiment of the invention;
Fig.6 shows a transform function with caching of an exemplary embodiment of the invention;
Fig.7 shows a sequence of two transform functions of an exemplary embodiment of the invention;
Fig.8 shows a sequence of two transform functions with caching of an exemplary embodiment of the invention;
Fig.9 shows a split key delivery process with caching in a receiver of an exemplary embodiment of the invention;
Fig.10 shows a receiver and a smartcard of an exemplary embodiment of the invention;
Fig. 11 shows two receivers sharing a smartcard of an exemplary embodiment of the invention;
Fig. 12 shows two receivers sharing a smartcard via a network of an exemplary embodiment of the invention;
Fig.13 shows a conditional access system of an exemplary embodiment of the invention;
Fig.14 shows the steps of a method for obtaining a control word in a receiver of an exemplary embodiment of the invention;
Fig.15 shows a diagram clarifying transformation functions and encryption in general terms;
Fig. 16A shows a block diagram of a function performing a mathematical transformation;
Fig. 16B shows a block diagram of a function performing a mathematical transformation under control of a seed;
Fig.17A shows a block diagram of an apply primitive;
Fig. 17B shows a block diagram of a remove primitive;
Fig. 17C shows a block diagram of a condition primitive;
Fig. 17D shows a block diagram of a combination of a remove and an apply primitive;
Fig. 17E shows a block diagram of a secure correlation of compounds; and
Fig.18 shows an illustrative example of a receiver applying transformation operations to obtain a control word.

DETAILED DESCRIPTION OF THE DRAWINGS

**[0036]** Caching is a known optimization technology in computer science that allows previously used data, in whatever form, to be stored and reused instead of being recomputed. As caches cannot be infinite in size, cached data is typically kept or discarded based on a usage pattern algorithm, such as e.g. a least recently used (LRU) algorithm, a most recently used (MRU) algorithm or a least-frequently used (LFU) algorithm.

**[0037]** A prior art example of an implementation of an entitlement transform tree is shown in Fig. 3. The invention enables caching of intermediate results in the entitlement transform tree, thereby increasing the computational efficiency. Caching functionality is created such that it can be implemented in the generic computation environment without affecting the tamper resistance of the conditional entitlement processing.

**[0038]** An intermediate value in the entitlement transform tree, e.g. $CWDK^{CTu2}$ shown in Fig.3, can be considered as non-sensitive data as it is only useful in the context of the sequence of functions and seeds in the entitlement transform tree. As a result, intermediate values can be stored in the generic computation environment for the purpose of caching, without degrading security.

**[0039]** Typically, intermediate data values that remain constant between two consecutive generations of frequently changing subkey are cached. It will be understood that caching is not limited to intermediate values for frequently changing subkeys and that intermediate values for less frequently changing subkeys can be cached as well. In case all intermediate values are unchanged between two consecutive generations of the CW itself, it is possible to have the resulting CW cached. In the latter case, typically the CW is cached in encrypted form, such as $\{CW\}_{CSSK}$.

**[0040]** Caching functionality is implemented by storing one or more of the intermediate values and/or end-result value together with an associated caching reference. Typically, the caching reference comprises input values, such as a transformed CW, a seed and/or a compound, and an indication of or reference to the function to which the input values are input for the calculation of the intermediate or end-result value.

**[0041]** The computation of the CW in the entitlement transform tree consists of a sequence of transform functions. A known example without caching is shown in Fig.3. Transform functions, such as shown in Figs. 1A-1E, are not stateful and typically conditional in that a semantically correct output value can be produced only when the inputs are correct.

**[0042]** Schematically each transform function can be represented as shown in Fig.4. Fig.4 shows transform function F that has two inputs a and b and that generates one output c. Inputs a and/or b and output c are protected by a mathematical transformation, enabling the data values x,y and z to be processed and stored in a untrusted domain such as a generic computation environment of a receiver.

**[0043]** Fig.5 shows a transformation function F with added caching functionality of an exemplary embodiment of the invention. In the embodiment of Fig.5 a cache control is added to the transformation function F. The cache transmits the input data values x and y to the inputs a and b, respectively, of the transform function F. The function F generates an output comprising the value z. Since the function is not stateful, the same set of input values x and y are always processed by transformation function F into the same output value z. The output z is stored in the cache to optimize later calculations of the same operation. The module implementing the transformation function has a cache control function that activates function F only when the resulting output value z for a given combination of inputs x and y is not

available in the cache. If the output value z for the given combination x and y is cached, then the cache provides the output value z to the output c directly.

**[0044]** Fig.6 shows a transformation function F with added caching functionality of an exemplary embodiment of the invention. In the embodiment of Fig.6 a cache control is added to the cache. The cache control transmits the input data values x and y to the inputs a and b, respectively, of the transform function F. The function F generates an output comprising the value z. Since the function is not stateful, the same set of input values x and y are always processed by transformation function F into the same output value z. The output z is stored in the cache to optimize later calculations of the same operation. The cache has a cache control function that activates function F only when the resulting output value z for a given combination of inputs x and y is not available in the cache. If the output value z for the given combination x and y is cached, then the cache provides the output value z to the output c directly.

**[0045]** In the examples of Fig.5 and Fig.6 the cache links function output values z with a set of function input parameter values x,y. Hereto a simple URL-style string may be used as a caching reference. Alternatively any other known data structure may be used to implement a caching reference. An example of a caching reference string is "Fc?Fa=x&Fb=y", describing the calculation of the result 'Fc' from transform function 'F' using the value 'x' for input 'Fa' and the value 'y' for the input 'Fb'. The caching reference "Fc?Fa=x&Fb=y" is stored in the cache along with the associated the function result 'z'.

**[0046]** As an example, the following table shows the cache entries as stored in the cache memory after the calculation of "F(x,y)=z" and "F(u,v) =w".

| Caching reference | Value |
|---|---|
| "Fc?Fa=x&Fb=y" | Z |
| "Fc?Fa=u&Fb=v" | W |

**[0047]** A next time the function F is activated, the cache first determines if the result for the calculation has been performed before. If there is a cache hit, the calculation is not carried out and the cached result is used instead.

**[0048]** Fig.7 shows an exemplary embodiment of the invention wherein a sequence of two transform functions produce an output y. Each of the transform functions F and G operates similar to the function F described in Fig.4 and is provided with caching functionality as described in Fig.5 or Fig.6.

**[0049]** In order to generate the output "Gc", the transform function G is activated with the value 'x' for its input parameter "Gb". The input parameter "Ga" is connected to the output of transform function "F(u,v)". The cache uses the result parameter string "Fc?Fa=u&Fb=v" to search for an earlier calculation of this function call. If the cache finds a result for this caching reference, the function 'F' does not need to be activated and the cached value w is used instead. The cache is then used to determine if the result of the calculation of "G(w,x)" is held in the cache. The cache now uses the caching reference "Gc?Ga=w&Gb=x" to search for the result. If found, the function 'G' does not need to be activated and the cached result 'y' is sent to the output of transform function 'G'. If the cache does not find a match, it activates the calculation of "G(w,x)". After the calculation, the result 'y' is returned to the cache and to the output "Gc". After these operations, the following cache entries are stored in the cache memory.

| Parameter name | Value |
|---|---|
| "Fc?Fa=x&Fb=y" | z |
| "Fc?Fa=u&Fb=v" | w |
| "Gc?Ga=w&Gb=x" | y |

**[0050]** The caching operation may be optimized by taking into account the structure of the transform tree and cache the combined result of a series of transform functions as a single string. This reduces access to the cache memory. E.g. in the example of Fig.7 using this optimization a single cache hit could produce the output value y instead of two cache hits. Hereto the cache content of the example of Fig.7 is extended with the following entry in the cache table.

| Parameter name | Value |
|---|---|
| "FGc?Fa=x&Fb=y&Gb=x" | z |

**[0051]** As shown in Fig. 5 and Fig.6, the cache control is implemented as a wrapper around a transform function or around a cache memory. In both implementations the cache control is configured to conditionally activate a transform

function module and provide the transform function with the relevant inputs.

**[0052]** Fig.8 shows an example of two transform functions F and G that are connected to a cache controller using a bus structure. The cache controller is connected to all transform function modules via the bus. This enables data u, v, w, x and y to be provided to the inputs Fa, Fb, Ga, Gb and outputs Fc, Gc, respectively, of the transform functions F and G. The cache control interface is used to activate the transform function modules in case an output value is not stored in the cache memory.

**[0053]** The cache controller of Fig.8 operates similar to the cache control showed in Fig.6. It will be understood that a bus structure as shown in Fig.8 can also be used with cache control functionality in each transform function F and G, similar to the operation of the cache control shown in Fig.5.

**[0054]** In addition to searching for cache entries and conditionally activating transform functions, the cache controller is optionally configured to remove unused cache entries to manage the memory size of the cache. Hereto the cache controller may use any known cache management technique to ensure that only the most relevant information is kept in the cache.

**[0055]** Fig.9 shows an example of an entitlement transform tree implementation extended with caching optimization. The transform functions in the entitlement transform tree of Fig.9 are similar to the transform functions of the entitlement transform tree shown in Fig.3 and the same transform function sequence is used. In the example of Fig.9 all data in the transform tree passes through the cache controller and a data interface indicated by "Data". The cache controller also controls the activation of the transform function modules via a cache control interface.

**[0056]** Fig.9 shows two parts of a receiver: a secure computation environment and a generic processing environment. The generic processing environment deals with external interfaces such as storage, data communication and user interaction. The secured computation environment deals with processing of keys and/or seeds. The processing is typically performed by one or more processors (not shown).

**[0057]** The ECM delivery path is used for the reception of entitlement control messages (ECM) from a head-end system. The ECM comprises an encrypted or transformed CW. The EMM delivery path is used for the reception of entitlement management messages (EMM) from the head-end system. The EMM comprises keys or seeds for decrypting or transforming the encrypted or transformed CW. The ECM delivery path and the EMM delivery path are typically implemented in an input module for receiving the ECMs and EMMs.

**[0058]** In the example of Fig.9 the generic computation environment contains the caching functionality, which is implemented as a cache controller and a cache memory. Via the cache controller data flows from the ECM delivery path and EMM delivery path to the cache memory and between the cache memory and the transform functions ($R_PA_{G1}$, $R_{G1}A_{U1}$ and $R_{G2}C_{vector}A_{U2}$) and TDES encryption whitebox.

**[0059]** In this example the following aliases are used for the transform functions: $F=R_PA_{G1}$, $G=R_{G1}A_{U1}$ and $H=R_{G2}C_{vector}A_{U2}$. F has to inputs Fa and Fb and an output Fc, G has two inputs Ga and Gb and an output Gc and H has two inputs Ha and Hb and an output Hc. All inputs and outputs of F, G and H are connected to the cache controller via the data bus.

**[0060]** Via the ECM delivery path a transformed control word in transformation space P, i.e. $CWD^{TP}$, is received. Via the EMM delivery path seeds <P, $G_1$>, <$G_1$, $U_1$>, <$G_2$, $U_2$, n> and <CSSK, $U_1$, $U_2$> are received. Via the EMM delivery path also a compound of a Control Word Difference Key CWDK in transformation space $G_2$ and a vector, i.e. <$CWDK^{TG2}$, vector>, is received for a group membership check.

**[0061]** The cache controller searches the cache memory for a cached output value of transform function F matching the input values being $CWD^{TP}$ for Fa and <P, $G_1$> for Fb. If the cached output value is found, the cached value is provided to Fc without invoking function F. If no output value is found, $CWD^{TP}$ is provided to Fa and <P, $G_1$> is provided to Fb via the data bus. Via the cache control interface an instruction is given from the cache controller to the transform function F to generate the output value using the input data on Fa and Fb and to return the result via Fc and the data bus to the cache controller. The result is stored in the cache memory, which now contains the following entry.

| Parameter name | Value |
|---|---|
| "Fc?Fa=$CWD^{TP}$&Fb=<P, $G_1$>" | $CWD^{TG1}$ |

**[0062]** Next, the cache controller searches the cache memory for a cached output value of transform function G matching the input values being the output value of F for Ga and <$G_1$, $U_1$> for Gb. If the cached output value is found, the cached value is provided to Gc without invoking function G. If no output value is found, the output value of F, in this example $CWD^{TG1}$, is provided to Ga and <$G_1$, $U_1$> is provided to Gb via the data bus. Via the cache control interface an instruction is given from the cache controller to the transform function G to generate the output value using the input data on Ga and Gb and to return the result via Gc and the data bus to the cache controller. The result is stored in the cache memory, which now contains the following entries.

| Parameter name | Value |
|---|---|
| "Fc?Fa=$CWD^{TP}$&Fb=<P, $G_1$>" | $CWD^{TG1}$ |
| "Gc?Ga=$CWD^{TG1}$&Gb=<$G_1$, $U_1$>" | $CWD^{TU1}$ |

**[0063]** Alternatively or optionally the result after processing the inputs by transform functions F and G is stored in the cache memory as a single entry, allowing the result of transform function G to be found in the cache memory in a single step using the input values Fa=$CWD^{TP}$, Fb=<P, $G_1$> and Gb=<$G_1$, $U_1$>. The cache memory then contains e.g. the following entries.

| Parameter name | Value |
|---|---|
| "Fc?Fa=$CWD^{TP}$&Fb=<P, $G_1$>" | $CWD^{TG1}$ |
| "Gc?Ga=$CWD^{TG1}$&Gb=<$G_1$, $U_1$>" | $CWD^{TU1}$ |
| "FGc?Fa=$CWD^{TP}$&Fb=<P, $G_1$>&Gb=<$G_1$, $U_1$>" | $CWD^{TU1}$ |

**[0064]** For the group membership check the cache controller searches the cache memory for a cached output value of secure correlation function H matching the input values being <$G_2$, $U_2$, n> for Ha and <$CWDK^{TG2}$, vector> for Hb. If the cached output value is found, the cached value is provided to Hc without invoking function H. If no output value is found, <$G_2$, $U_2$, n> is provided to Ha and <$CWDK^{TG2}$, vector> is provided to Hb via the data bus. Via the cache control interface an instruction is given from the cache controller to the secure correlation function H to generate the output value using the input data on Ha and Hb and to return the result via Hc and the data bus to the cache controller. The result is stored in the cache memory, which now contains the following entries.

| Parameter name | Value |
|---|---|
| "Fc?Fa=$CWD^{TP}$&Fb=<P, $G_1$>" | $CWD^{TG1}$ |
| "Gc?Ga=$CWD^{TG1}$&Gb=<$G_1$, $U_1$>" | $CWD^{TU1}$ |
| "FGc?Fa=$CWD^{TP}$&Fb=<P, $G_1$>&Gb=<$G_1$, $U_1$>" | $CWD^{TU1}$ |
| "Hc?Ha=<$G_2$, $U_2$, n>&Hb=<$CWDK^{TG2}$, vector>" | $CWDK^{CTU2}$ |

**[0065]** In a last step the output data of Gc and Hc, i.e. $CWD^{TU1}$ and $CWDK^{CTU2}$, respectively, are provided together with seed <CSSK, $U_1$, $U_2$> to the TDES Encryption Whitebox via the data bus for the generation of $\{CW\}_{CSSK}$. The resulting $\{CW\}_{CSSK}$ is typically not cached in the cache memory to prevent this data from being obtained in the generic computation environment.

**[0066]** The implementation of the transform tree in the secure computation environment is not limited to the example of Fig.9. There are typically two or more transform functions in a sequence of transform functions. The transform functions can form an entitlement transform tree with one or more branches. Each transform function can be any known transform function. Instead of the TDES encryption whitebox any other encryption function may be used for the generation of an encrypted control word from a transformed control word. Alternatively, instead of the TDES encryption whitebox a remove primitive may be used to generate a clear text control word from a transformed control word.

**[0067]** Caching may not be efficient for all steps in the transform sequence. For example, when a cache hit ratio is expected to be low for a particular transform function, caching functionality can actually reduce overall processing performance due to cache memory access prior to performing the transform function. To avoid such reduction of overall processing performance, one or more transform function modules can be implemented without caching functionality, resulting in its input values being processed by the transform function module for the generation of the output value without searching for a cache hit.

**[0068]** In the example of Fig.9 the generic computation environment and the secure computation environment are parts of a receiver. Alternatively, the secure computation environment is implemented in a smartcard and the generic computation environment is implemented in a receiver.

**[0069]** Instead of a smartcard having a traditional form factor, any other computing device implementing smartcard technology may be used as a smartcard, such as e.g. a PC running smartcard emulation software.

**[0070]** Fig.10 shows a simplified architecture containing a set-top box (STB) as a digital TV receiver and a smartcard

that is communicatively connected to the STB, e.g. through insertion of the smartcard in the STB. EMMs and ECMs received by the STB can be stored in an EMM/ECM storage before processing by the smartcard. The smartcard obtains data from the ECMs and EMMs in any manner known per se, which data includes the input data for an entitlement transform tree in the secure computation environment of the smartcard. Inputs to and outputs from transform functions and/or secure correlation functions are stored in the cache in the STB.

**[0071]** It is possible to have two or more networked devices share a common smartcard. In the example of Fig.11 two STBs each store ECMs and EMMs received from a headend system in encrypted form in a non-volatile memory indicated by EMM/ECM storage. STABs without an inserted smartcard establish a secure connection to the smartcard via a network and through the intermediary of the STB wherein the smartcard is inserted. The smartcard receives one or more ECMs and related EMMs from the receiver and decrypts the ECMs and EMMs to obtain the input data for the transform function modules in the secure computation environment of the smartcard. Output values of the transform functions are transmitted via the secure connection to the STB for storage in a local cache of the STB.

**[0072]** An alternative to the example of Fig.11 is shown in Fig.12, wherein the smartcard is communicatively connected to the network instead of inserted in one of the STBs, and wherein each STB accesses the smartcard via the network.

**[0073]** A STB typically has more storage space than a smartcard. Therefore, in the examples of Figs.10-12 the cache memory is implemented in the STB. Alternatively it is possible to implement the cache memory in the smartcard.

**[0074]** Fig.13 shows a conditional access system of an exemplary embodiment of the invention. A head-end system transmits ECMs and EMMs to one or more receivers via the distribution network. The ECM typically contains the transformed control word, e.g. $CWD^{TP}$ of Fig.9, which is to be processed by the entitlement transform tree in the secure computation environment of the receiver. The secure computation environment may be implemented in a smartcard that is communicatively connected to the receiver. The EMM typically contains one or more seeds, e.g. $<P, G_1>$ and $<G_1, U_1>$ of Fig.9, used in the transformation of the transformed control word. Other data, such as group membership check data, may be transmitted in the EMM as well. Multiple EMMs may be used for the transmission of the data.

**[0075]** In Fig.14 the steps performed by a receiver of an exemplary embodiment of the invention are schematically shown. In step 101 the transformed control word and the seed are received. In step 102 the transformed control word and the seed are intercepted in the cache control module. In step 103 the control word matching the transform function, the transformed control word and the seed is searched in the cache memory. In step 104 the result of the search is analysed. If the control word was found, it is provided to an output of the transform function in step 105 to thereby bypass the transform function. If the control word was not found, it is provided to the transform function together with the seed in step 106. In step 107 the transformed control word is migrated from an input transform space to an output transform space to obtain the control word using a mathematical transformation under control of the seed. In step 108 the control word is stored in the cache memory associatively with the transform function, the transformed control word and the seed. Step 108 enables a cache hit in step 103 a next time the same transform function is called with the same input values.

**[0076]** As discussed above, data and software obfuscation techniques can make use of transformation functions to obfuscate intermediate results. The concept of transformation functions, as used in this disclosure differs from encryption. The differences are further clarified in general with reference to FIG.15 and the discussion that follows below.

**[0077]** Assume, there exists an input domain ID with a plurality of data elements in a non-transformed data space. An encryption function E using some key is defined that is configured to accept the data elements of input domain ID as an input to deliver a corresponding encrypted data element in an output domain OD. By applying a decryption function D, the original data elements of input domain ID can be obtained by applying the decryption function D to the data elements of output domain OD. In a non-secure environment (typically referred to as "white-box"), an adversary is assumed to know input and output data elements and have access to internals of encryption function E during execution. Unless extra precautions are taken in this environment, secrets (e.g., the key used in encryption/decryption functions) can be derived easily by an adversary.

**[0078]** Additional security can be obtained in a non-secured environment by applying transformation functions to the input domain ID and output domain OD, i.e. the transformation functions are input- and output operations. Transformation function $T_1$ maps data elements from the input domain ID to transformed data elements of transformed input domain ID' of a transformed data space. Similarly, transformation function $T_2$ maps data elements from the output domain OD to the transformed output domain OD'. Transformed encryption and decryption functions E' and D' can now be defined between ID' and OD' using transformed keys. In case inverse transformations are to be performed, e.g. when results are to be communicated to the non-transformed space, $T_1$ and $T_2$ are bijections.

**[0079]** Using transformation functions $T_1$, $T_2$, together with encryption techniques implies that, instead of inputting data elements of input domain ID to encryption function E to obtain encrypted data elements of output domain OD, transformed data elements of domain ID' are inputted to transformed encryption function E' by applying transformation function $T_1$. Transformed encryption function E' combines the inverse transformation function $T_1^{-1}$ and the transformation function $T_2$ in the encryption operation to protect the confidential information, such as the key. Then transformed encrypted data elements of domain OD' are obtained. Keys for encryption functions E or decryption function D cannot be retrieved when analyzing input data and output data in the transformed data space. This ensures that, even when operating in a

non-secure environment, the keys are protected against adversaries. In particular, transformations enables systems to never reveal any part of the key, or any value derived from it, in the clear in contiguous memory. In other words, transformation obfuscates data by applying transformations and operating on the data in the transformed space. In some embodiments, these transformation functions are randomly generated.

**[0080]** An advantage of using transformations to obfuscate data allows the input and output values of these transformations to be stored or cached in the generic (non-secure) computation environment, due to the characteristic that these input and output values are useless to an adversary. In contrast, input and output values of non-transformed encryption functions cannot be stored or cached in the generic computation environment. These values, such as encrypted and decrypted/cleartext key pairs, should not be cached and stored in non-secure memory because they may be useful to an adversary.

**[0081]** One of the transformation functions $T_1$, $T_2$ should be a non-trivial function. In case, $T_1$ is a trivial function, the input domains ID and ID' are the same domain. In case, $T_2$ is a trivial function, the output domains are the same domain.

**[0082]** The function F shown in Fig.16A is a mathematical operation that migrates data Z across two different transform spaces identified by IN and OUT. The dimension of the output transform space OUT is at least as large as the input transform space IN, and any data Z is represented (possibly not uniquely) in both input and output transform spaces as X and Y respectively. The function F is designed such that it is difficult to run in reverse direction. Because no apparent mapping between the input and output transform spaces exists and the dimension of transform spaces IN and OUT is preferably significantly large, recreation of the function F is prevented. Moreover, the function F is implemented in such a way that it is difficult to extract the data Z as it passes through the function, e.g. using white-box techniques and/or other code obfuscation techniques.

**[0083]** With reference to Fig.16A, function F is e.g. defined as Y=F(X)=3*X+2. If the input transform space IN is a clear text transform space, then $X=(Z)^{IN}=Z$. After migration the following result is obtained: $Y=(Z)^{OUT}=3*X+2$. To migrate Z from the output transform space to the clear text transform space again, a reverse function $F^{-1}(Y)=(Y-2)/3$ must be available to obtain X as follows: $F^{-1}(Y)=(3*X+2-2)/3=X$. In this example Z, X and Y are numbers that can be used to transform using simple addition and subtraction mathematics. It is to be understood that Z, X and Y can be data in any data format, including binary values, numbers, characters, words, and etcetera. The function F can be a more complex function and suitable for operation on e.g. binary values, numbers, characters or words. In some embodiments, the function F is chosen in a manner that is computationally efficient to implement on binary values.

**[0084]** The function F can be defined as a mathematical operation that can be seeded with an additional parameter S, as shown in Fig.16B. The migration that the function F performs is typically defined by the seed S. This type of seeded transform functions are used in relation to Typically, no information about the input space IN and output space OUT is embedded into F. The function F is chosen such that manipulation of input data X or seed S yields an unpredictable resulting data Y in the output transform space. The seed S does not need to be secured or stored in a secure environment as the seed S is engineered in such a way that no information about transform space IN or OUT can be extracted.

**[0085]** With reference to Fig.16B, function F is e.g. defined as F(X,S)=X-7+S. If the input transform space IN is a clear text transform space, then $X=(Z)^{IN}=Z$. After migration the following result is thus obtained: $Y=(Z)^{OUT}=X-7+S=Z-7+S$. If e.g. a seed S is provided as data comprising the value of 5, then F(X,5)=X-7+5 and $Y=(Z)^{OUT}=X-7+5=Z-2$. To migrate Z from the output transform space to the clear text transform space again, a reverse function $F^{-1}(Y,S)=Y+7-S$ must be available to obtain X as follows: $F^{-1}(Y,S)=(X-7+5)+7-S$. If the seed S=5 is known, then Z can correctly be obtained as: $F^{-1}(Y,5)=(X-7+5)+7-5=X=Z$.

**[0086]** If the input transform space IN is not a clear text transform space, then function F typically first performs a reverse transformation in the input transform space IN and next a transformation in the output transform space OUT. Such function F is e.g. defined as $F(X,S_1,S_2)=F_2(F_1^{-1}(X,S_1),S_2)$, wherein $F_1^{-1}(X,S_1)=X-2-S_1$ and $F_2(X,S_2)=X-7+S_2$. After migration the following result is thus obtained: $Y=(Z)^{OUT}=(X-2-S_1)-7+S_2=X-9-S_1+S_2$, wherein $X=(Z)^{IN}$.

**[0087]** Seeds $S_1$ and $S_2$ can be provided as two separate seeds to first perform $F_1^{-1}(X,S_1)$ and next perform $F_2(X,S_2)$, or more preferably as a compound of seeds $<S_1,S_2>$. Generally, a compound of seeds is a mixture of multiple seeds. From the mixture of multiple seeds the individual seeds are not derivable. A parameter mixing function for mixing the seeds $S_1$ and $S_2$ is denoted as: $f(S_1,S_2)=<S_1,S_2>$. The function result $<S_1,S_2>$ is called the compound of seeds $S_1$ and $S_2$. In the example above, if $S_1=5$ and $S_2=7$, then one compound is $<S_1, S_2>=5-7=-2$.

**[0088]** In the above examples Z, X, Y and S are numbers that can be used to transform using simple addition and subtraction mathematics. It will be understood that Z, X, Y and S can be data in any data format, including binary values, numbers, characters, words, and etcetera. The function F can be a more complex function and suitable for operation on e.g. binary values, numbers, characters or words. Similar to Figs.1A-E, Figs.17A describes the basic primitives in further detail.

**[0089]** In Fig.17A the function $A(Data,S)=A_s(Data)=Data^{TS}$ defines an apply primitive that transforms an input Data into a transformed $Data^{TS}$ using an input seed S. In Fig.17B the function $R(Data^{TS},S)=R_S(Data^{TS})$ =Data defines a remove primitive that reverses the transformation of an input $Data^{TS}$ using a seed S to obtain an output Data. The seed S need to be identical for the two functions A() and R() to become the inverse of each other.

**[0090]** The original Data and its transformed variant $Data^{TS}$ are typically of a same size, i.e. represented by a same number of bits, making it impossible to determine, based on its size, whether or not the Data is in a particular transformed space.

**[0091]** In Fig. 17C the function $C(Data_1,Data_2)=C_{Data1}(Data_2)=Data^C$ defines a conditional transformation wherein the output $Data^C$ is a correlation of the two inputs $Data_1$ and $Data_2$. The condition primitive typically preserves the size of the input data and output data, making it impossible to determine whether or not the Data is the result of a correlation.

**[0092]** Primitives such as the apply primitive, remove primitive and the condition primitive can be combined. The combination produces a new operation wherein the individual primitives are invisible.

**[0093]** Fig.17D shows an example of a combination of a remove and an apply primitive. The transformation operation uses a compound <P,S> as input to the combined remove and apply operation applied to input $Data^{TP}$. The $R_PA_S$ function maps the input $Data^{TP}$ from input transform domain P to output transform domain S to obtain output $Data^{TS}$. All inputs and outputs of the combined remove and apply operation are either transformed or in the form of a compound. The operation is applied to transformed data and produces transformed data. Thus the transformation operation takes place in transformed domain spaces and reveals no individual parameters or untransformed data on any of the interfaces. The function used to produce the compound <P,S> is preferably unique and linked to the implementation of the combined apply and remove operation.

**[0094]** Fig.17E shows an example of a secured correlation operation on two input compounds <P, S, $Q_1$> and <$Data^{TP}$, Q2>. The $R_PC_QA_S$ function combines a remove, condition and apply primitive to thereby create output $Data^{CTS}$.

**[0095]** Fig.18 shows an illustrative example of a transformation path implemented in a receiver of a conditional access receiver. The receiver is typically implemented at least partly as software or as a field-programmable gate array (FPGA) program in a programmable array. The receiver comprises an unprotected, partially protected and/or secure memory of a processor. The processor is configured to execute functions stored in the memory to migrate a secret data Z from an input transform space IN to an output transform space OUT. The secret data Z cannot be extracted or intercepted and thus cannot e.g. be illegally distributed to other receivers.

**[0096]** The receiver receives a control word CW as a globally transformed control word $CWD^T_P$ in an entitlement control message ECM. The receiver migrates the CWD from the input transform space P into the final output transform space CSSK of the receiver in three steps. The last migration step creates a transformed control word $\{CW\}_{CSSK}$, which is the control word CW in the output transform space of the cluster shared secret key (CSSK) that is unique to the receiver.

**[0097]** The receiver comprises a generic computation environment and a secure computation environment.

**[0098]** The generic computation environment comprises an ECM Delivery Path for receiving the ECM from the head-end system. The generic computation environment further comprises an EMM Delivery Path for receiving an Entitlement Management Messages (EMM) from the head-end system. The EMM comprises the seeds that are used to migrate $CWD^T_P$ through the various transform spaces along the path of the transformation path. The seeds received in the EMM are stored in a NVRAM memory of the generic computation environment. A first seed equals the compound <P,$G_1$>. A second seed equals the compound <$G_1$,$U_1$>. A third seed equals the compound <CSSK, $U_1$>.

**[0099]** The secure computation environment comprises a sequence of transformation functions. A first function $R_pA_{G1}$ transforms $CWD^T_P$ from the input transform space P to the output transform space $G_1$ using the compound <P,$G_1$> as seed input. Subsequently a second function $R_{G1}A_{U1}$ transforms $CWD^T_{G1}$, i.e. the CW in the transform space $G_1$, from the input transform space $G_1$ to the output transform space $U_1$ using the compound <$G_1$, $U_1$>. Subsequently a third function, in this example a TDES Whitebox Encryption function, transforms $CWD^T_{U1}$, i.e. the CW in the transform space $U_1$, from the input transform space $U_1$ to the output transform space CSSK. The resulting $\{CW\}_{CSSK}$ is the CW encrypted under the CSSK key, which can be decrypted by the conditional access receiver using the CSSK that is pre-stored in a secured memory of the receiver or securely derivable by the receiver.

**[0100]** It is to be understood that the transformation path in the receiver is not limited to the example shown in Fig.18 and may contain any number and any kind of transformation operations.

**[0101]** One embodiment of the disclosure may be implemented as a program product for use with a computer system. The program(s) of the program product define functions of the embodiments (including the methods described herein) and can be included on a variety of computer-readable non-transitory storage media. The computer-readable storage media can be a non-transitory storage medium. Illustrative computer-readable and/or non-transitory storage media include, but are not limited to: (i) non-writable storage media (e.g., read-only memory devices within a computer such as CD-ROM disks readable by a CD-ROM drive, flash memory, ROM chips or any type of solid-state non-volatile semiconductor memory) on which information is permanently stored; and (ii) writable storage media (e.g., floppy disks within a diskette drive or hard-disk drive or any type of solid-state random-access semiconductor memory) on which alterable information is stored. Moreover, the disclosure is not limited to the embodiments described above, which may be varied within the scope of the accompanying claims without departing from the scope of the disclosure.

**[0102]** It is to be understood that any feature described in relation to any one embodiment may be used alone, or in combination with other features described, and may also be used in combination with one or more features of any other of the embodiments, or any combination of any other of the embodiments. Furthermore, equivalents and modifications

not described above may also be employed without departing from the scope of the invention, which is defined in the accompanying claims.

## Claims

1. A receiver for securely obtaining a control word, the receiver comprising a first memory configured for storing a transform function (F) configured to receive a transformed control word (x) and a seed (y) and to migrate the transformed control word (x) from an input transform space to an output transform space to obtain the control word (z) using a mathematical transformation under control of the seed (y), the receiver further comprising a cache memory and a cache control module, wherein the cache control module is configured to:

   intercept the transformed control word (x) and the seed (y);
   search in the cache memory for the control word (z) matching the transform function (F), the transformed control word (x) and the seed (y);
   if the control word (z) is found in the cache memory, provide the control word (z) to an output of the transform function (F) thereby bypassing the transform function (F); and
   if the control word (z) is not found in the cache memory, provide the control word (x) and the seed (y) to the transform function (F), obtain the control word (z) from the transform function (F) and store the control word (z) associatively with the transform function (F), the transformed control word (x) and the seed (y) in the cache memory.

2. The receiver according to claim 1, wherein the first memory is configured for storing two or more transform functions (F,G) forming a sequence of transform functions and/or a tree of transform functions, and wherein the cache control module is linked to a data bus communicatively connecting the cache memory with one or more of the transform functions (F,G).

3. The receiver according to claim 2, wherein a last transform function in the sequence of transform functions and/or tree of transform functions is configured to generated the control word as a clear text control word or an encrypted control word.

4. The receiver according to any one of the claims 1-3, comprising a secure computation environment comprising the first memory, the receiver further comprising a generic computation environment comprising the cache memory.

5. The receiver according to claim 4, wherein the generic computation environment comprises the cache control module and wherein the data bus communicatively connects the cache control module to inputs and the output of the one or more of the transform functions (F,G).

6. The receiver according to claim 4, wherein each of the one or more of the transform functions comprises a cache control module and wherein the data bus communicatively connects each cache control module to the cache memory.

7. The receiver according to any one of the claims 4-6, further comprising a smartcard, the smartcard possibly being detachably connected to the receiver, and wherein the smartcard comprises the secure computation environment.

8. The receiver according to claim 7, wherein the receiver is communicatively linked to the smartcard via a network.

9. A smartcard for use in a receiver according to claim 7 or claim 8, the smartcard comprising a first memory in a secure computation environment, the first memory being configured for storing a transform function (F) configured to receive a transformed control word (x) and a seed (y) and to migrate the transformed control word (x) from an input transform space to an output transform space to obtain a control word (z) using a mathematical transformation under control of the seed (y), the transform function (F) comprising a cache control module configured to:

   intercept the transformed control word (x) and the seed (y);
   search in a cache memory of the receiver for the control word (z) matching the transform function (F), the transformed control word (x) and the seed (y);
   if the control word (z) is found in the cache memory, provide the control word (z) to an output of the transform function (F) thereby bypassing the transform function (F); and
   if the control word (z) is not found in the cache memory, provide the control word (x) and the seed (y) to the

transform function (F), obtain the control word (z) from the transform function (F) and store the control word (z) associatively with the transform function (F), the transformed control word (x) and the seed (y) in the cache memory.

**10.** The smartcard according to claim 9, wherein the first memory is configured for storing two or more transform functions (F,G) forming a sequence of transform functions and/or a tree of transform functions, and wherein the cache control module is linked to a data bus communicatively connecting the cache memory with one or more of the transform functions (F,G).

**11.** A conditional access system comprising a head-end system and one or more receivers according to any one of the claims 1-8, the head-end system being configured to transmit an entitlement control message comprising the transformed control word (x) and an entitlement management message comprising one or more seeds (y) to the receiver.

**12.** A method for securely obtaining a control word in a receiver comprising a first memory configured for storing a transform function (F), the method comprising the steps of:

receiving (101) a transformed control word (x) and a seed (y);
intercepting (102) in a cache control module the transformed control word (x) and the seed (y);
searching (103) in a cache memory for the control word (z) matching the transform function (F), the transformed control word (x) and the seed (y);
if (104) the control word (z) is found in the cache memory, providing (105) the control word (z) to an output of the transform function (F) thereby bypassing the transform function (F); and
if (104) the control word (z) is not found in the cache memory, providing (106) the control word (x) and the seed (y) to the transform function (F), migrating (107) in the transform function (F) the transformed control word (x) from an input transform space to an output transform space to obtain the control word (z) using a mathematical transformation under control of the seed (y), and storing (108) the control word (z) associatively with the transform function (F), the transformed control word (x) and the seed (y) in the cache memory.


Fig.1A
(prior art)


Fig.1B
(prior art)

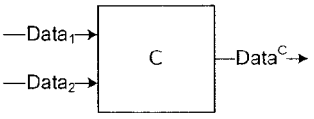


Fig.1C
(prior art)

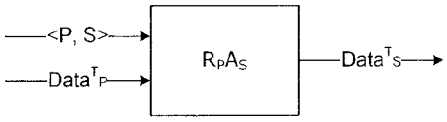


Fig.1D
(prior art)

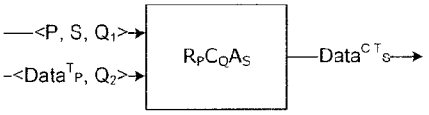


Fig.1E
(prior art)

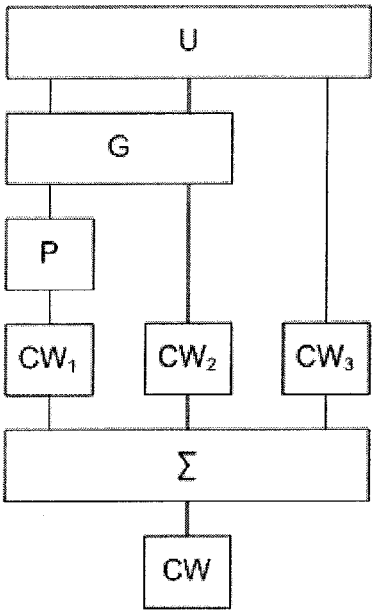


Fig.2
(prior art)

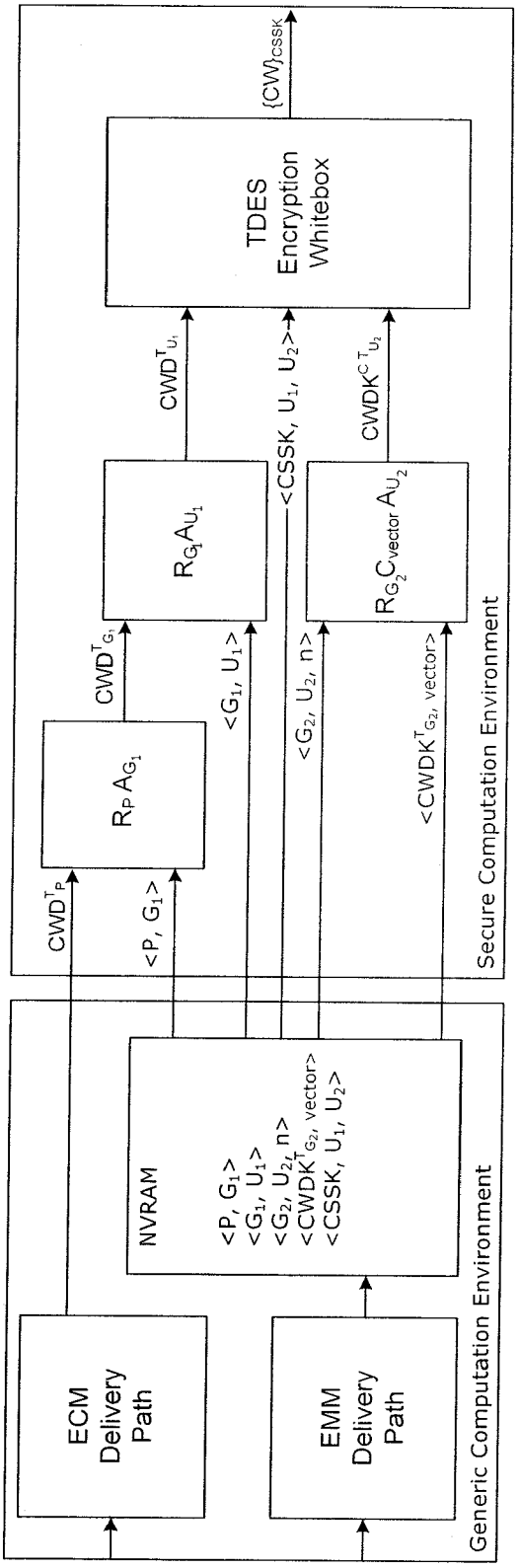


Fig.3
(prior art)

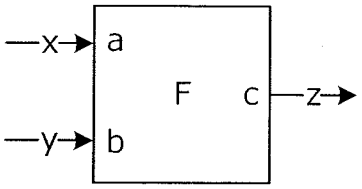


Fig.4

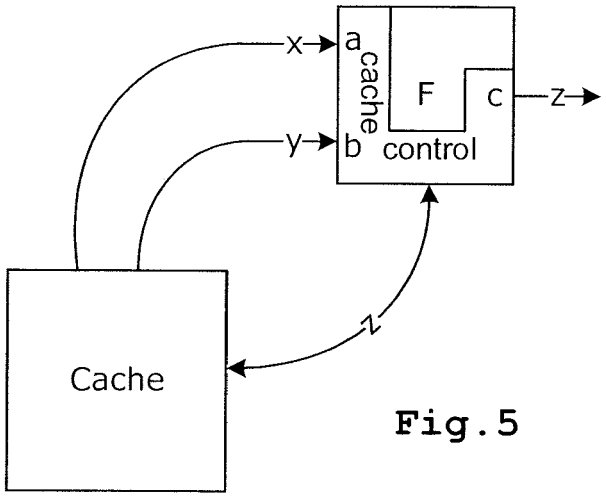


Fig.5

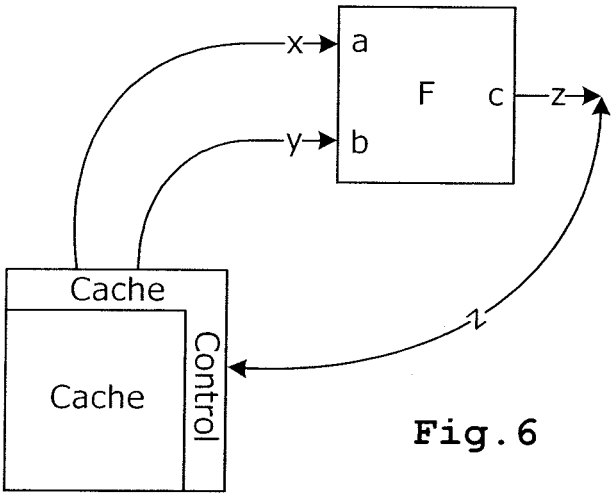


Fig.6

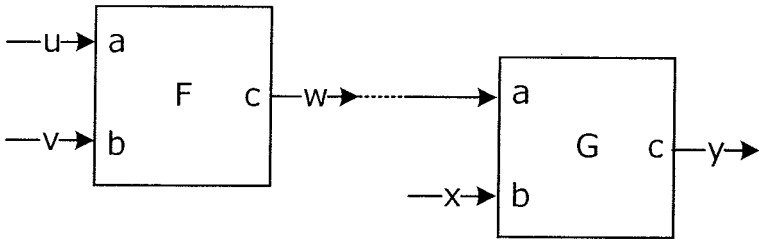


Fig.7

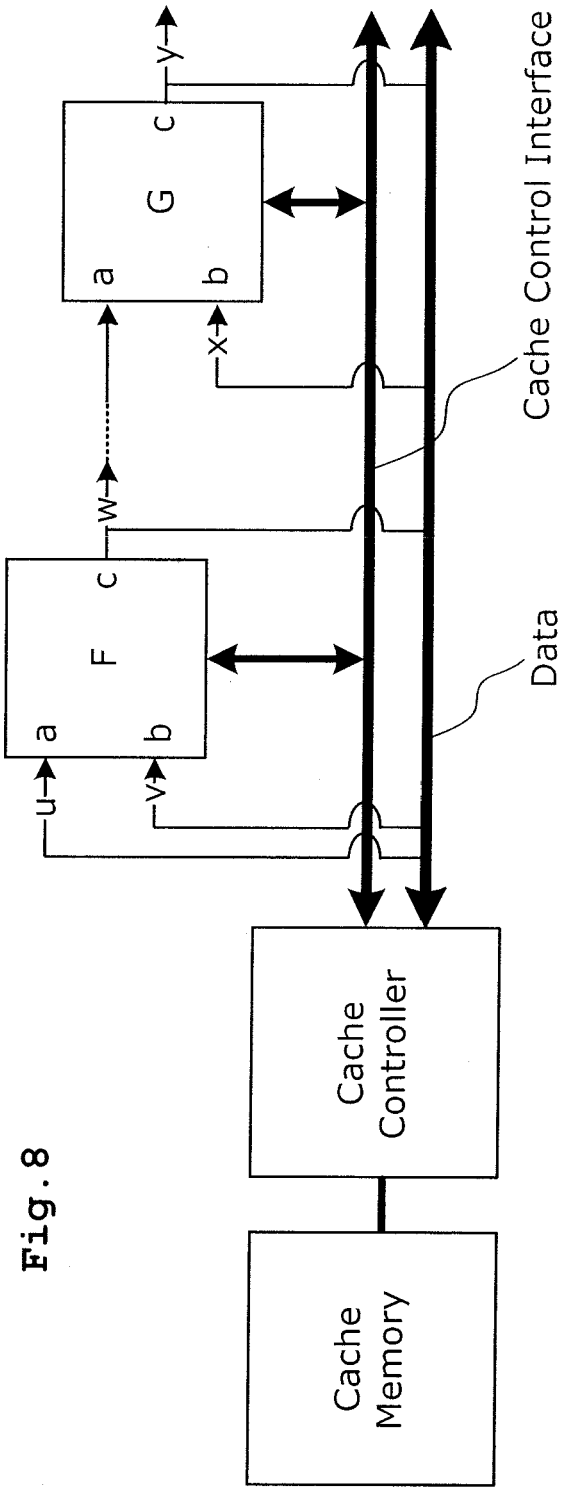
Cache Control Interface
Data
Cache Controller
Cache Memory
F
G
a
b
c
u
v
w
x
y

Fig.8

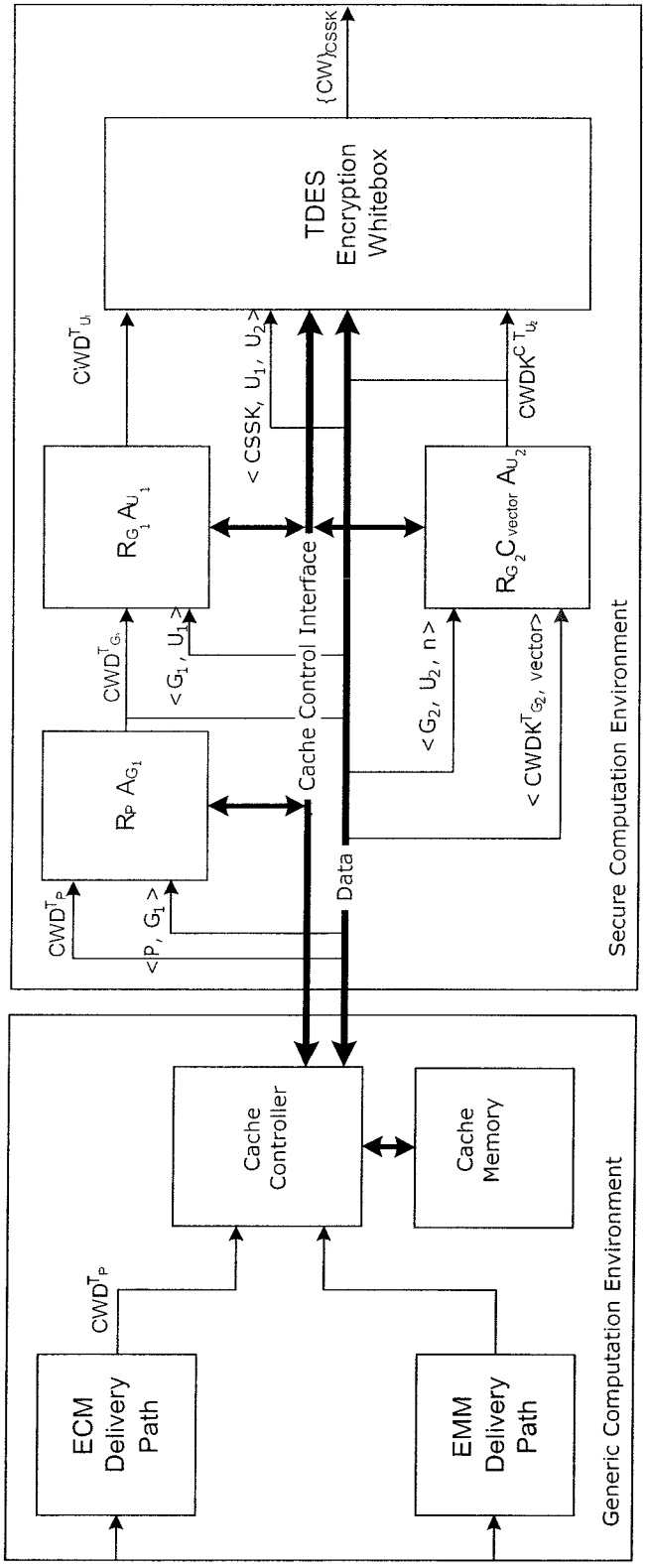
{CW}CSSK
TDES Encryption Whitebox
CWD^T_Ui
< CSSK, U1, U2>
CWDK^CT_Ut
R_G1 A_U1
R_G2 C_vector A_U2
CWD^T_G1
< G1, U1>
< G2, U2, n>
< CWDK^T_G2, vector>
Cache Control Interface
R_P A_G1
CWD^T_P
<P, G1>
Data
Secure Computation Environment
Cache Controller
Cache Memory
CWD^T_P
ECM Delivery Path
EMM Delivery Path
Generic Computation Environment

Fig.9

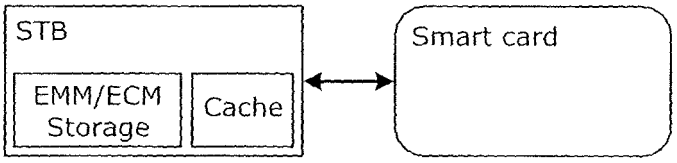


Fig.10

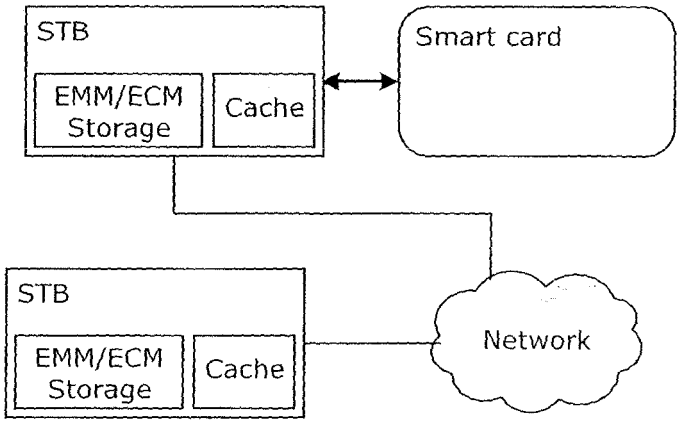


Fig.11

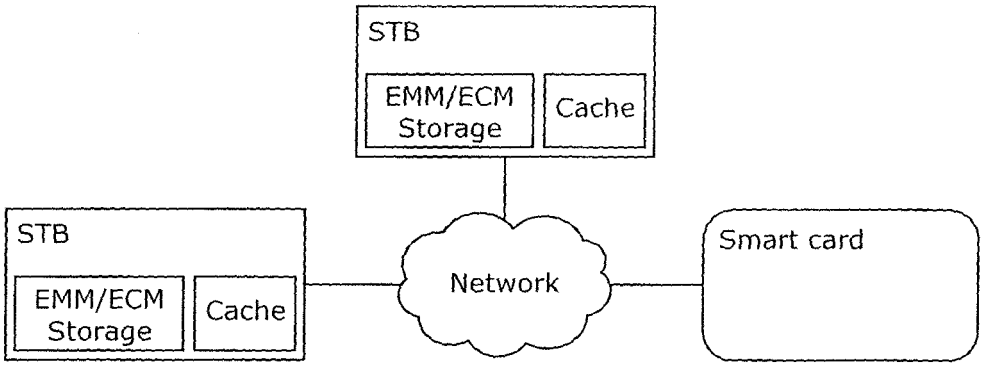


Fig.12

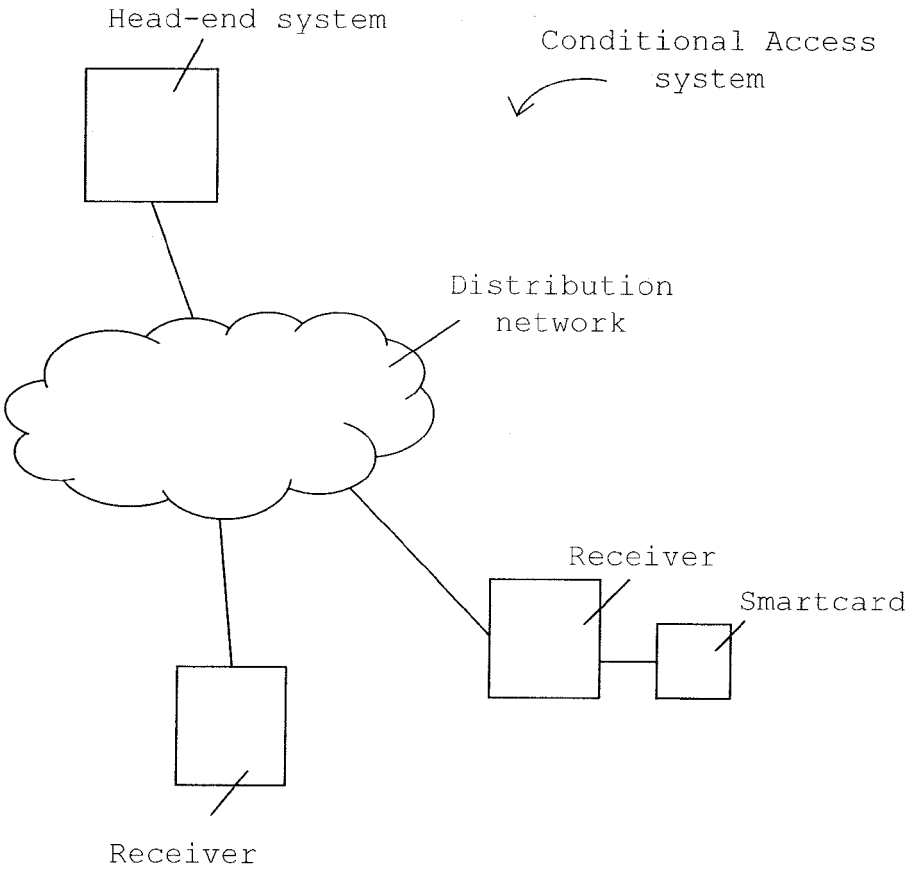
Head-end system
Conditional Access system
Distribution network
Receiver
Smartcard
Receiver

Fig.13

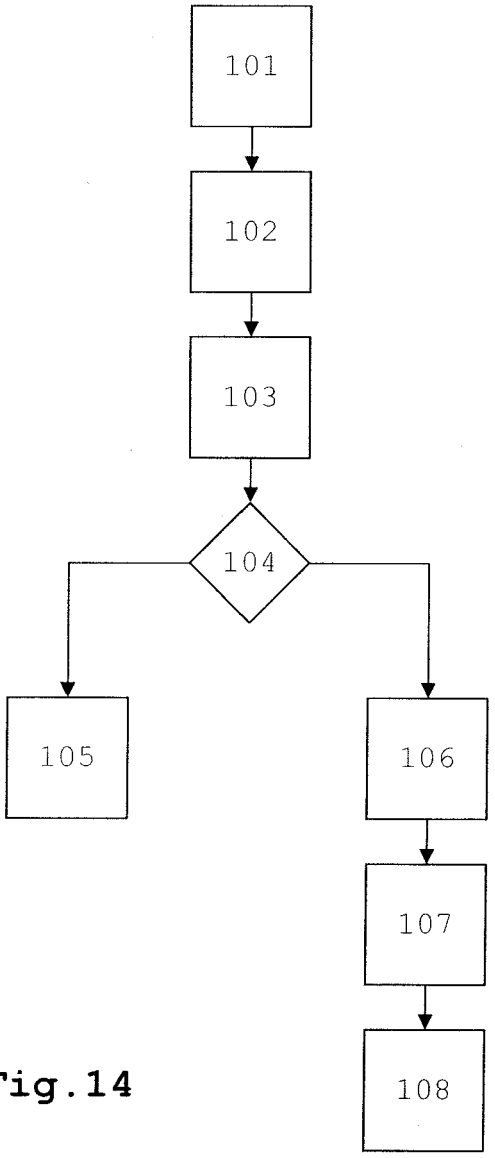


Fig.14

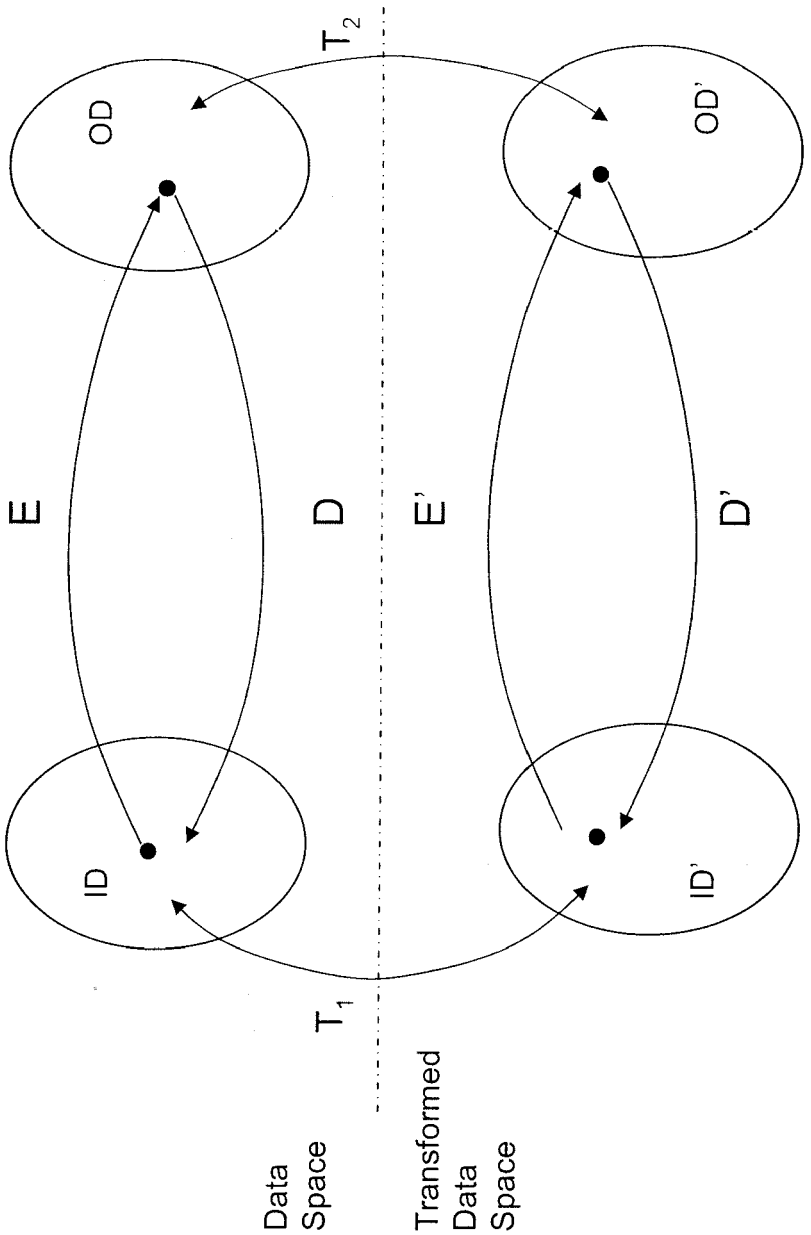


Fig.15

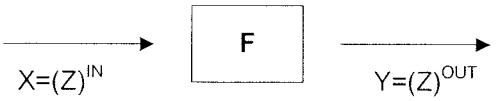


Fig.16A

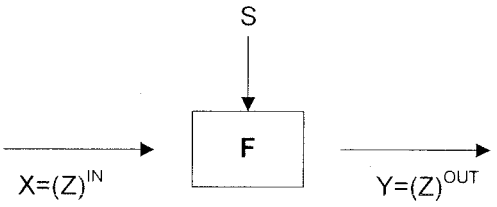


Fig.16B

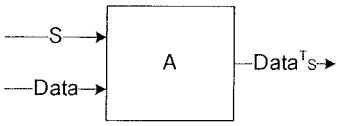


Fig.17A

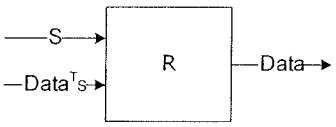


Fig.17B

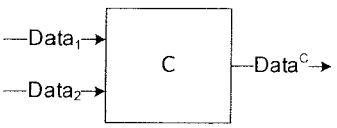


Fig.17C

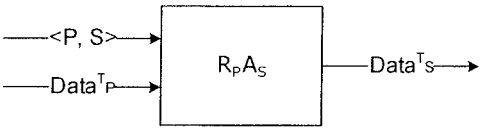


Fig.17D

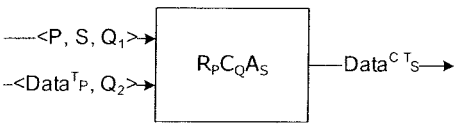


Fig.17E

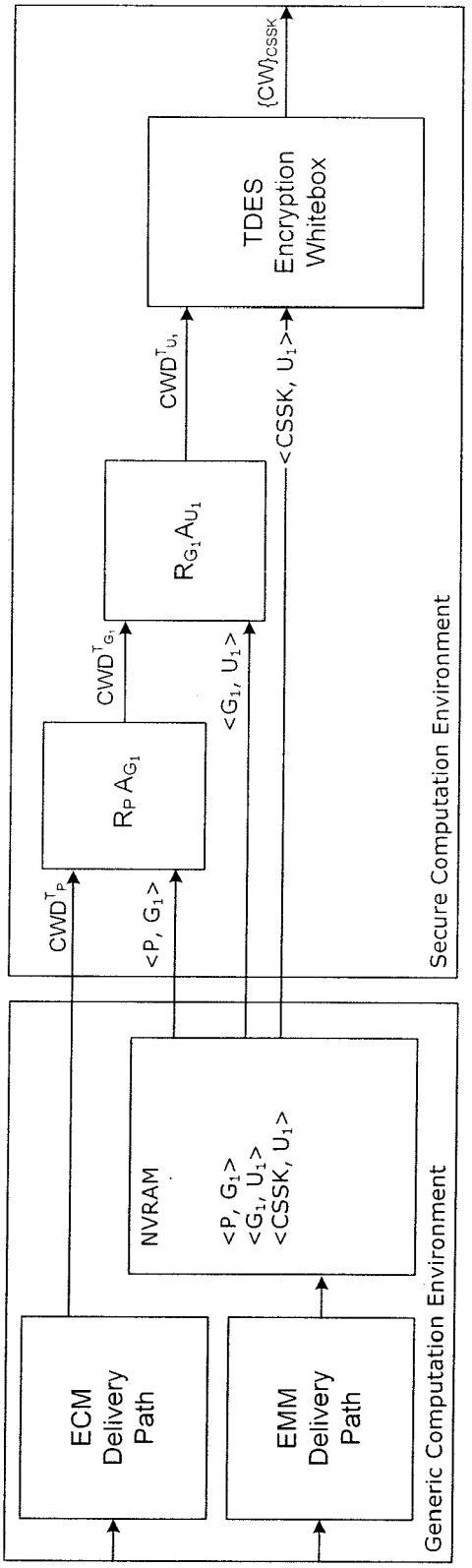
Generic Computation Environment
ECM Delivery Path
EMM Delivery Path
NVRAM
<P, G1>
<G1, U1>
<CSSK, U1>
Secure Computation Environment
CWD^T_P
<P, G1>
RP AG1
CWD^T_G1
<G1, U1>
RG1 AU1
CWD^T_U1
<CSSK, U1>
TDES Encryption Whitebox
{CW}CSSK

Fig.18

Europäisches Patentamt
European Patent Office
Office européen des brevets

# EUROPEAN SEARCH REPORT

Application Number
EP 10 19 6221

## DOCUMENTS CONSIDERED TO BE RELEVANT

| Category | Citation of document with indication, where appropriate, of relevant passages | Relevant to claim | CLASSIFICATION OF THE APPLICATION (IPC) |
|---|---|---|---|
| Y | US 2004/086127 A1 (CANDELORE BRANT L [US]) 6 May 2004 (2004-05-06) * paragraphs [0048] - [0053], [0061] - [0064], [0115], [0116]; figures 2,4,5,12 * | 1-12 | INV. H04N7/16 H04L9/08 |
| Y | HENNESSY J L ET AL: "Computer Architecture. A quantative approach, passage", COMPUTER ARCHITECTURE: A QUANTITATIVE APPROACH, XX, XX, 1 June 2002 (2002-06-01), pages 390-423, XP002318184, * pages 390,393; figure 5.1 * | 1-12 | |
| A | WO 2006/091304 A2 (COMCAST CABLE HOLDINGS LLC [US]; FAHRNY JAMES WILLIAM [US]; COMPTON CH) 31 August 2006 (2006-08-31) * pages 11-17; figures 1,2 * | 1-12 | |
| A | EP 1 220 541 A2 (SONY CORP [JP]) 3 July 2002 (2002-07-03) * abstract * * pages 8,9 * | 1-12 | TECHNICAL FIELDS SEARCHED (IPC) H04N H04L |

The present search report has been drawn up for all claims

| Place of search | Date of completion of the search | Examiner |
|---|---|---|
| Munich | 10 March 2011 | Folea, Octavian |

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

## ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 10 19 6221

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

10-03-2011

| Patent document cited in search report | Publication date | Patent family member(s) | Publication date |
|---|---|---|---|
| US 2004086127 A1 | 06-05-2004 | US 2009168996 A1 | 02-07-2009 |
| | | US 2004088558 A1 | 06-05-2004 |
| | | US 2004088552 A1 | 06-05-2004 |
| WO 2006091304 A2 | 31-08-2006 | CA 2598747 A1 | 31-08-2006 |
| | | EP 1851712 A2 | 07-11-2007 |
| | | US 2006200412 A1 | 07-09-2006 |
| EP 1220541 A2 | 03-07-2002 | CA 2366161 A1 | 28-06-2002 |
| | | CN 1362810 A | 07-08-2002 |
| | | JP 2002261747 A | 13-09-2002 |
| | | KR 20020055353 A | 08-07-2002 |
| | | SG 114516 A1 | 28-09-2005 |
| | | TW 548938 B | 21-08-2003 |

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

## REFERENCES CITED IN THE DESCRIPTION

*This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.*

### Patent documents cited in the description


- EP 09155007 A **[0008] [0009] [0010]**